data.table_note–Efficient reshaping using data.tables

王泽贤

2016 年 *12* 月 *7* 日

简介

在 data.tables中，reshape2包中的对应函数被扩展为 melt和 dcast。

在本章中，我们要做的是：

1. 简短的介绍如何用 data.table中默认的 melting和 casting函数来将数据在 wide和 long形式之间转换。
2. 通过一个场景实现，认识现在函数的笨重和低效。
3. 最后学习如何用新的 melt和 dcast方法来高效处理数据塑性问题。

注意：如果你需要使用同时使用 *data.table* 和 *reshape2* 的函数，请确保 *data.table* 在 *reshpe2* 之后加载

1. 默认功能

```
library(data.table)
```

a) melting (wide to long)

假设我们有一个人造的数据：

```
DT = fread('melt_default.csv')
DT
```

```
##    family_id age_mother dob_child1 dob_child2 dob_child3
## 1:         1         30 1998-11-26 2000-01-29         NA
## 2:         2         27 1996-06-22         NA         NA
## 3:         3         26 2002-07-11 2004-04-05 2007-09-02
## 4:         4         32 2004-10-10 2009-08-27 2012-07-21
## 5:         5         29 2000-12-05 2005-02-28         NA
```

```
str(DT)
```

```
## Classes 'data.table' and 'data.frame':   5 obs. of  5 variables:
##  $ family_id : int  1 2 3 4 5
##  $ age_mother: int  30 27 26 32 29
##  $ dob_child1: chr  "1998-11-26" "1996-06-22" "2002-07-11" "2004-10-10" ...
##  $ dob_child2: chr  "2000-01-29" NA "2004-04-05" "2009-08-27" ...
##  $ dob_child3: chr  NA NA "2007-09-02" "2012-07-21" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

这是一个 wide形式的数据

- 转换 DT为 long形式数据，使得每个 dob列（变量）的名字，变成一个新变量 variable中的值。
- 原本的同一个样本，按照 dob列，扩展为多个样本。

```
DT.m1 <- melt(DT, id.vars = c('family_id', 'age_mother'),
        measure.vars = c('dob_child1', 'dob_child2', 'dob_child3'))
DT.m1
```

```
##    family_id age_mother   variable      value
## 1:         1         30 dob_child1 1998-11-26
## 2:         2         27 dob_child1 1996-06-22
## 3:         3         26 dob_child1 2002-07-11
## 4:         4         32 dob_child1 2004-10-10
## 5:         5         29 dob_child1 2000-12-05
## 6:         1         30 dob_child2 2000-01-29
## 7:         2         27 dob_child2         NA
## 8:         3         26 dob_child2 2004-04-05
## 9:         4         32 dob_child2 2009-08-27
## 10:        5         29 dob_child2 2005-02-28
## 11:        1         30 dob_child3         NA
## 12:        2         27 dob_child3         NA
## 13:        3         26 dob_child3 2007-09-02
## 14:        4         32 dob_child3 2012-07-21
## 15:        5         29 dob_child3         NA
```

```
str(DT.m1)
```

```
## Classes 'data.table' and 'data.frame':   15 obs. of  4 variables:
##  $ family_id : int  1 2 3 4 5 1 2 3 4 5 ...
##  $ age_mother: int  30 27 26 32 29 30 27 26 32 29 ...
##  $ variable  : Factor w/ 3 levels "dob_child1","dob_child2",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ value     : chr  "1998-11-26" "1996-06-22" "2002-07-11" "2004-10-10" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

说明

1. measure.vars 指定了我们需要合并的变量
2. 我们也可以直接指定哪几列，而不是繁琐地输入名字，如 measure.vars = c(3,4,5)
3. 默认情况下，合并后的新变量 variable是 factor, 可以在函数中附加 variable.factor=FALSE来返回字符变量 (只有在加载了 data.table包后才有这个功能)
4. 默认情况下，合并后的列会被放入 variable和 value中
5. id.vars是需要保留的变量 ‘

- 为合并后产生的新变量命名

```
DT.m1 <- melt(DT, measure.vars = c('dob_child1', 'dob_child2', 'dob_child3'),
        variable.name = 'child', value.name = 'dob')
DT.m1
```

```
##    family_id age_mother      child        dob
## 1:         1         30 dob_child1 1998-11-26
## 2:         2         27 dob_child1 1996-06-22
## 3:         3         26 dob_child1 2002-07-11
## 4:         4         32 dob_child1 2004-10-10
## 5:         5         29 dob_child1 2000-12-05
## 6:         1         30 dob_child2 2000-01-29
## 7:         2         27 dob_child2         NA
## 8:         3         26 dob_child2 2004-04-05
## 9:         4         32 dob_child2 2009-08-27
## 10:        5         29 dob_child2 2005-02-28
## 11:        1         30 dob_child3         NA
## 12:        2         27 dob_child3         NA
## 13:        3         26 dob_child3 2007-09-02
## 14:        4         32 dob_child3 2012-07-21
## 15:        5         29 dob_child3         NA
```

说明

1. 默认情况下，如果 id.vars没有指定，那么程序会自动把 measure.vars之外剩下的所有变量都放进去。反之亦然。
2. 如果 id.vars和 measure.vars都没有指定，那么所有非数值、整数、逻辑列都会被放入 id.vars中。并且此时会有一个警告，提示你自动放入的变量是哪些

b) Casting (long to wide)

在前面的小节中，我们看到了如何把 wide形式的数据转化为 long形式。这一节我们要学习如何反向转化

- 如何从 DT.m转化回原始的 DT

我们希望把具有同样的 family_id和 age_mother的数据的不同 child合并到同一行：

```
dcast(DT.m1, family_id + age_mother ~ child, value.var = 'dob')
```

```
##    family_id age_mother dob_child1 dob_child2 dob_child3
## 1:         1         30 1998-11-26 2000-01-29         NA
## 2:         2         27 1996-06-22         NA         NA
## 3:         3         26 2002-07-11 2004-04-05 2007-09-02
## 4:         4         32 2004-10-10 2009-08-27 2012-07-21
## 5:         5         29 2000-12-05 2005-02-28         NA
```

说明

1. dcast 中的公式用法：~左边是要保留的变量 (id vars)，右边是要转换的变量 (measure vars)。
2. value.var是用来填充到每一行的数据列名。

- 利用 DT.m，如何获得每个家庭中的孩子数

将计算函数传入 fun.aggregate(简写 fun.agg)。

```
dcast(DT.m1, family_id ~ . ,fun.agg = function(x) sum(!is.na(x)), value.var = 'dob')
```

```
##    family_id .
## 1:         1 2
## 2:         2 1
## 3:         3 3
## 4:         4 3
## 5:         5 2
```

说明

1. family_id ~ . 表示用 family_id作为分组计算变量
2. fun.agg = function(x) sum(!is.na(x)) 表示对传入的 x，计算不是 NA的个数
3. value.var是传入 fun.agg中的 x

2.melt/dcast 方法的限制

前面所诉的 melt和 dcast方法基于 reshape2包，但是已经经过 data.table的改进，效率更高

但是在某些情形下，我们会发现我们想要的需求无法用一个直接的语法来实现，比如：

```
DT <- fread('melt_enhanced.csv')
DT
```

```
##    family_id age_mother dob_child1 dob_child2 dob_child3 gender_child1
## 1:         1         30 1998-11-26 2000-01-29         NA            1
## 2:         2         27 1996-06-22         NA         NA            2
```

```
## 3:         3        26 2002-07-11 2004-04-05 2007-09-02            2
## 4:         4        32 2004-10-10 2009-08-27 2012-07-21            1
## 5:         5        29 2000-12-05 2005-02-28         NA            2
##    gender_child2 gender_child3
## 1:             2            NA
## 2:            NA            NA
## 3:             2             1
## 4:             1             1
## 5:             1            NA
```

```
## 1 = female, 2 = male
```

如果你需要把用 melt把所有的 dob列合并成一列，gender列合并成一列，用原来的方法：

```
DT.m1 = melt(DT, id = c("family_id", "age_mother"))
```

```
## Warning in melt.data.table(DT, id = c("family_id", "age_mother")):
## 'measure.vars' [dob_child1, dob_child2, dob_child3, gender_child1, ...]
## are not all of the same type. By order of hierarchy, the molten data value
## column will be of type 'character'. All measure variables not of type
## 'character' will be coerced to. Check DETAILS in ?melt.data.table for more
## on coercion.
```

```
DT.m1[, c("variable", "child") := tstrsplit(variable, "_", fixed = TRUE)]
```

```
##     family_id age_mother variable      value  child
##  1:         1         30      dob 1998-11-26 child1
##  2:         2         27      dob 1996-06-22 child1
##  3:         3         26      dob 2002-07-11 child1
##  4:         4         32      dob 2004-10-10 child1
##  5:         5         29      dob 2000-12-05 child1
##  6:         1         30      dob 2000-01-29 child2
##  7:         2         27      dob         NA child2
##  8:         3         26      dob 2004-04-05 child2
##  9:         4         32      dob 2009-08-27 child2
## 10:         5         29      dob 2005-02-28 child2
## 11:         1         30      dob         NA child3
## 12:         2         27      dob         NA child3
## 13:         3         26      dob 2007-09-02 child3
## 14:         4         32      dob 2012-07-21 child3
## 15:         5         29      dob         NA child3
## 16:         1         30   gender          1 child1
## 17:         2         27   gender          2 child1
## 18:         3         26   gender          2 child1
## 19:         4         32   gender          1 child1
## 20:         5         29   gender          2 child1
## 21:         1         30   gender          2 child2
## 22:         2         27   gender         NA child2
## 23:         3         26   gender          2 child2
## 24:         4         32   gender          1 child2
## 25:         5         29   gender          1 child2
## 26:         1         30   gender         NA child3
## 27:         2         27   gender         NA child3
## 28:         3         26   gender          1 child3
## 29:         4         32   gender          1 child3
## 30:         5         29   gender         NA child3
```

```
##     family_id age_mother variable      value  child
```

```
DT.c1 = dcast(DT.m1, family_id + age_mother + child ~ variable, value.var = "value")
DT.c1
```

```
##     family_id age_mother  child        dob gender
##  1:         1         30 child1 1998-11-26      1
##  2:         1         30 child2 2000-01-29      2
##  3:         1         30 child3         NA     NA
##  4:         2         27 child1 1996-06-22      2
##  5:         2         27 child2         NA     NA
##  6:         2         27 child3         NA     NA
##  7:         3         26 child1 2002-07-11      2
##  8:         3         26 child2 2004-04-05      2
##  9:         3         26 child3 2007-09-02      1
## 10:         4         32 child1 2004-10-10      1
## 11:         4         32 child2 2009-08-27      1
## 12:         4         32 child3 2012-07-21      1
## 13:         5         29 child1 2000-12-05      2
## 14:         5         29 child2 2005-02-28      1
## 15:         5         29 child3         NA     NA
```

```
str(DT.c1)
```

```
## Classes 'data.table' and 'data.frame':   15 obs. of  5 variables:
##  $ family_id : int  1 1 1 2 2 2 3 3 3 4 ...
##  $ age_mother: int  30 30 30 27 27 27 26 26 26 32 ...
##  $ child     : chr  "child1" "child2" "child3" "child1" ...
##  $ dob       : chr  "1998-11-26" "2000-01-29" NA "1996-06-22" ...
##  $ gender    : chr  "1" "2" NA "2" ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr  "family_id" "age_mother" "child"
```

你会发现 gender变成字符串了

问题

1. 我们只是想把 dob和 gender列各自合并，却需要全部合并在一起，再分开。
2. 要合并的列可能是不同的数据类型，一开始全部合并后，会被强制转换。
3. 操作繁琐，计算量大

3. 强化版的 (新) 功能

a) 加强的 melt

- 同时使用 melt合并多个行

想法很简单，我们向 measure.vars中传入由多个列组成的 list，这些列就是我们想要合并的列

```
colA <-  paste('dob_child', 1:3, sep = "")
colB <-  paste('gender_child', 1:3, sep = "")
DT.m2 <- melt(DT, measure = list(colA, colB), value.name = c('dob', 'gender'))
DT.m2
```

```
##     family_id age_mother variable       dob gender
##  1:         1         30        1 1998-11-26      1
##  2:         2         27        1 1996-06-22      2
##  3:         3         26        1 2002-07-11      2
```

```
##  4:         4        32        1 2004-10-10      1
##  5:         5        29        1 2000-12-05      2
##  6:         1        30        2 2000-01-29      2
##  7:         2        27        2         NA     NA
##  8:         3        26        2 2004-04-05      2
##  9:         4        32        2 2009-08-27      1
## 10:         5        29        2 2005-02-28      1
## 11:         1        30        3         NA     NA
## 12:         2        27        3         NA     NA
## 13:         3        26        3 2007-09-02      1
## 14:         4        32        3 2012-07-21      1
## 15:         5        29        3         NA     NA
```

说明

1. measure传入要合并的所有列的列名，用 list 的不同元素区分分组
2. value.name 指定合并后产生的新变量列名
3. 额外生成的 variable是指的是这些数据来源于 list中第一个变量组的第几个变量

```
str(DT.m2)
```

```
## Classes 'data.table' and 'data.frame':   15 obs. of  5 variables:
##  $ family_id : int  1 2 3 4 5 1 2 3 4 5 ...
##  $ age_mother: int  30 27 26 32 29 30 27 26 32 29 ...
##  $ variable  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 2 2 2 2 2 ...
##  $ dob       : chr  "1998-11-26" "1996-06-22" "2002-07-11" "2004-10-10" ...
##  $ gender    : int  1 2 2 1 2 2 NA 2 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

可以看到这样操作的时候，列的原始数据类型都会保留。

- 使用 patterns()

有时候我们需要合并的变量的名字有一定的规律，那么就可以用 pattern()函数，可以通过正则表达式来指定要合并的列名：

```
DT.m2 <- melt(DT, measure = patterns('^dob','^gender'), value.name = c('dob', 'gender'))
DT.m2
```

```
##     family_id age_mother variable        dob gender
##  1:         1        30        1 1998-11-26      1
##  2:         2        27        1 1996-06-22      2
##  3:         3        26        1 2002-07-11      2
##  4:         4        32        1 2004-10-10      1
##  5:         5        29        1 2000-12-05      2
##  6:         1        30        2 2000-01-29      2
##  7:         2        27        2         NA     NA
##  8:         3        26        2 2004-04-05      2
##  9:         4        32        2 2009-08-27      1
## 10:         5        29        2 2005-02-28      1
## 11:         1        30        3         NA     NA
## 12:         2        27        3         NA     NA
## 13:         3        26        3 2007-09-02      1
## 14:         4        32        3 2012-07-21      1
## 15:         5        29        3         NA     NA
```

这些功能全部以 C 语言重写，速度和效率都有很大进步。

b) 加强的 dcast

现在我们已经可以方便地把多个列同时各自合并。现在给定 DT.m2，如何转换回去？

如果我们用之前的 dcast的功能，那么我们必须拆分 2 次再合并结果。存在的问题和之前的 melt一样。

- 同时拆分多个列

```
DT.c2 <- dcast(DT.m2, family_id + age_mother ~ variable, value.var = c('dob','gender'))
DT.c2
```

```
##    family_id age_mother      dob_1      dob_2      dob_3 gender_1 gender_2
## 1:         1         30 1998-11-26 2000-01-29         NA        1        2
## 2:         2         27 1996-06-22         NA         NA        2       NA
## 3:         3         26 2002-07-11 2004-04-05 2007-09-02        2        2
## 4:         4         32 2004-10-10 2009-08-27 2012-07-21        1        1
## 5:         5         29 2000-12-05 2005-02-28         NA        2        1
##    gender_3
## 1:       NA
## 2:       NA
## 3:        1
## 4:        1
## 5:       NA
```

说明

1. 按照 variable分组拆分，用 value.var内的列为填充值。如 variable为 1 的 dob为一组取出来
2. 同样可以在其中使用 fun.agg函数来进行计算如

```
dcast(DT.m2, family_id + age_mother ~ variable, fun.agg = function(x) sum(!is.na(x)), value.var = c('dob','gender'))
```

```
##    family_id age_mother dob_1 dob_2 dob_3 gender_1 gender_2 gender_3
## 1:         1         30     1     1     0        1        1        0
## 2:         2         27     1     0     0        1        0        0
## 3:         3         26     1     1     1        1        1        1
## 4:         4         32     1     1     1        1        1        1
## 5:         5         29     1     1     0        1        1        0
```

也就是说如果没有指定 fun.agg，那么则用 value.var的列的原始值作为填充，如果有指定 fun.agg那么就要会输出 fun.agg计算后的内容。本例中即变成求每种符合条件的非 NA 个数。